F0693P-US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Title : AN APPARATUS FOR ANALYZING SOFTWARE AND METHOD OF THE SAME

Inventor(s) : Hiroaki KIMURA
Masahiro NOGUCHI

発明の名称：ソフトウェア解析装置および方法

【開示の要約】（Abstract of the Disclosure）

コンピュータプログラムを解析してプログラム解析情報を自動的に作成するプログラム解析情報作成手段と、作成されたプログラム解析情報を適当な単位ごとに所定のデータ記録媒体に順次格納するプログラム解析情報格納手段とを備え、プログラム解析情報作成手段によって作成される複数種類のプログラム解析情報を、個々の解析情報が得られるごとに順次データベース化してデータ記録媒体に格納していくようにすることにより、大規模なプログラムに対してプログラム解析情報を得ようとした場合に、コンピュータに実装できるメモリ容量に制限があっても、解析の途中でメモリ不足などが発生することもなく、目的とするプログラム解析情報を確実に得ることができるようにする。

【発明の背景】（Background of the Invention）

［発明の分野］（Field of the Invention）

本発明は、コンピュータプログラムの内容を理解しやすくするために、コンピュータプログラムについて解析を行うソフトウェア解析装置および方法、更にはこれらの機能を実現させるためのプログラムを格納した記録媒体に関する。

［関連技術の説明］（Description of the Related Art）

多数のプログラマが協力して大規模なプログラムを開発する場合や、既にあるプログラムの保守を行おうとする場合など、他人が書いたプログラムを理解しなければならないことがある。しかし、他人が書いたプログラムを理解することには、大きな困難を伴うことが多い。また、プログラムを開発する過程でそれまでに自分が開発した部分を見直したりすることもある。ところが、プログラムが大規模になると、自分が書いたプログラムでさえも、見直し等に多くの労力を要していた。

そこで、従来、プログラムのソースコードを自動的に解析して様々な情報を提供することにより、プログラムの理解を支援するための装置が提案されている。その一例として、プログラムのソースコードを解析してコールグラフやフローグラフを作成し、プログラムを視覚的に把握できるようにする装置および方法がある。これについては、例えば、本出願人が既に出願した特願平９－３２４１５号、特願平９－３２４５２号などに開示されている。

なお、「コールグラフ」とは、プログラムを構成する各手続き（Ｃ言語では「関数」）の間の呼び出し関係を示したグラフである。このグラフのノードは手続きを表し、エッジは手続き間の呼び出し関係を表している。このコールグラフをグラフィック表示させると、手続き間の呼び出し関係が視覚的に把握できるだけでなく、プログラムの構造化の程度や内容の理解、あるいは望ましくない手続き呼び出しの検出が容易になる。

また、「フローグラフ」とは、ある一つの手続き内における制御の流れ（フロー）を示したグラフである。このグラフのノードは基本ブロック（連続した文の

列からなり、制御が先頭の文に移された後は、途中で停止したり分岐したりせず、最後の文まで制御が到達するものを言う）を表し、エッジは基本ブロック間の制御のフローを表す。このフローグラフについても、その情報をグラフィック表示させると、その手続き内の制御フローが視覚的に把握できるだけでなく、望ましくない制御フローの検出が容易となる。

しかしながら、大規模なプログラム、例えば１００万ステップを超えるようなプログラムに対して解析を行おうとすると、プログラム解析を行うために必要な解析情報のボリュームも非常に大きくなる。このため、通常のワークステーションやパーソナルコンピュータが実装できる最大容量のメモリ（ワークメモリとして使われる主記憶装置）内に全ての解析情報が入りきらないとか、解析に要する時間が非常に長くなるといった不都合が発生する。

例えば、解析途中にメモリ容量が不足し、解析情報がそれ以上メモリに入らなくなってしまった場合には、解析はそれ以上先に進めることはできない。この場合は、解析対象のプログラムを幾つかに分割したり作り直したりするなど、何らかの対応策を講じた上で、最初からプログラムの解析を再度行わなければならず、作業の効率を悪くしていた。

また、解析対象のプログラムを分割すると、プログラムの全体に渡って行う必要がある解析、例えば、プログラム中のある変数の値を変えたときにそれが他のどの部分に影響を与えているかを検出する影響解析などでは、適切な解析情報が得られなくなってしまうという問題も生じる。

さらに、プログラムの解析を行っている際に、メモリ不足以外の何らかの要因で、解析が中断されてしまうこともある。この場合、中断されるまでに行われた解析が全て無駄になり、プログラムの解析を最初からやり直さなければならず、作業の効率を悪くしていた。

【発明の要約】（Summary of the Invention）

本発明は上記事情に基づいて成されたものであり、その目的は、コンピュータに実装できるメモリ容量に制限がある場合でも、大規模なプログラムの解析を確実に行うことができるようにすることにある。

また、本発明の他の目的は、プログラムの解析を行っている際に何らかの要因で解析が中断された場合等でも、中断される間際までに得られた解析情報を有効に利用することができるようにすることにある。

上記の目的を達成するために、本発明に係るソフトウェア解析装置は、コンピュータプログラムの解析に必要なプログラム解析情報を自動的に作成するプログラム解析情報作成手段と、上記プログラム解析情報作成手段によって作成された上記プログラム解析情報を、適当な単位もしくは適当なタイミングごとに所定のデータ記録媒体に順次格納するプログラム解析情報格納手段と、上記データ記録媒体から上記プログラム解析情報を読み出してプログラム解析を実行するプログラム解析手段とを備えたことを特徴とする。

また、本発明に係るソフトウェア解析方法は、コンピュータプログラムの解析に必要なプログラム解析情報を自動的に作成するプログラム解析情報作成ステップと、上記プログラム解析情報作成ステップにて作成された上記プログラム解析情報を、適当な単位もしくは適当なタイミングごとに所定のデータ記録媒体に順次格納するプログラム解析情報格納ステップと、上記データ記録媒体から上記プログラム解析情報を読み出してプログラム解析を実行するプログラム解析ステップとを有することを特徴とする。

また、本発明に係るコンピュータ読み取り可能な記録媒体は、コンピュータプログラムの解析に必要なプログラム解析情報を自動的に作成するプログラム解析情報作成機能と、上記プログラム解析情報作成機能によって作成された上記プログラム解析情報を、適当な単位もしくは適当なタイミングごとに所定のデータ記録媒体に順次格納するプログラム解析情報格納機能と、上記データ記録媒体から上記プログラム解析情報を読み出してプログラム解析を実行するプログラム解析機能と、をコンピュータに実行させるためのプログラムを記録したことを特徴とする。

【図面の簡単な説明】（Brief Description of the Drawings）

図１は、本発明によるソフトウェア解析装置の一実施例について説明するため図である。

図２は、本実施例によるソフトウェア解析装置がデータベースに格納する解析情報の構造の一例を示す図である。

図３は、本実施例のプログラム解析処理部について詳しく説明するための図である。

図４は、本実施例によるソフトウェア解析装置の動作、つまり本実施例によるソフトウェア解析方法の手順を示すフローチャートである。

【好適実施例の詳細な説明】

(Detailed Description of the Preferred Embodiments)

以下に、本発明の実施例を図面を参照しながら説明する。図１は本発明によるソフトウェア解析装置の一実施例について説明するための図、図２は本実施例によるソフトウェア解析装置がデータベースに格納する解析情報の構造の一例を示す図、図３は本実施例のプログラム解析処理部を詳しく説明するための図、図４は本実施例によるソフトウェア解析装置の動作、つまり本実施例によるソフトウェア解析方法の手順を示すフローチャートである。

図１において、ソースファイル１０は、所定の言語（例えばＣ言語）で記述されたプログラムのソースコードが記録されたファイルである。このソースファイル１０は、プログラム解析情報作成部１１に読み込まれる。プログラム解析情報作成部１１は、ユーザからの指示（コンピュータとの対話入力や設定ファイルの入力などの様々な方法）により、プログラム解析を行うために必要となる所定のプログラム解析情報を作成する処理を行う。

プログラム解析情報格納部１２は、例えばプログラム解析情報作成部１１により個々のプログラム解析情報が作成される毎に、ハードディスク等の情報記録媒体１３上のデータベースに順次格納する。すなわち、１つのプログラム解析情報が作成される度に、そのことがプログラム解析情報作成部１１からプログラム解析情報格納部１２に伝えられる。そして、これを受けたプログラム解析情報格納部１２が、そのとき作成されたプログラム解析情報を情報記録媒体１３に格納する。

そして、プログラム解析処理を行う場合に、プログラム解析処理部１５が必要とするプログラム解析情報をプログラム解析情報読出部１４が情報記録媒体１３から取り出し、それをプログラム解析処理部１５に供給する。

また、以前のプログラム解析に必要とするプログラム解析情報が少なかったり、解析情報の作成中に中断が起こったりして、プログラム解析情報が情報記録媒体１３にある程度入っている場合は、情報記録媒体１３に何が入っているかという情報と、その解析情報とをプログラム解析情報読出部１４が取り出して、それ

らの情報をもとにプログラム解析情報作成部１１が必要なプログラム解析情報を作成するということになる。

以上に述べたプログラム解析情報作成部１１、プログラム解析情報格納部１２、プログラム解析情報読出部１４およびプログラム解析処理部１５は、実際にはＣＰＵ、ＲＯＭおよびＲＡＭ等から成るコンピュータにより構成されている。当該コンピュータは、プログラム解析情報作成部１１により作成されたプログラム解析情報を情報記録媒体１３に移す前に一時的に蓄えておくための主記憶装置（以下、メモリと呼ぶ）を備えている。

本実施例の場合、プログラム解析情報には、図２に示すように、構文解析木（シンボルテーブルを含む）、コールグラフ、フローグラフ、データフロー情報、プログラム依存グラフ、モジュール入出力情報、メトリクス情報、冗長度情報および保守用文書情報などが含まれる。これらのうち、構文解析木、コールグラフ、フローグラフ、データフロー情報、プログラム依存グラフ、モジュール入出力情報などは、図１のプログラム解析情報作成部１１が作成し、メトリクス情報、冗長度情報および保守用文書情報などは、プログラム解析をバッチ処理で行う図３のプログラム解析部２０；が作成する。

ここで、「シンボルテーブル」とは、プログラムに用いられているシンボル（変数）の意味等を表した情報の表である。例えば、プログラム中の異なる場所で同じ「ａ」という文字を使っている場合でも、それらが同じ変数を指す場合と、異なる変数を指す場合とがある。例えば、異なる二つの関数の中で同じ文字の変数が使われている場合、その変数が大域変数であれば、同じものであるし、それぞれの関数においてローカルに定義された変数であれば、別々の変数となる。

したがって、プログラム中の異なる場所に同じ文字で記述された変数が複数あったときに、それらが同じものなのか異なるものなのかは、直ちには識別できない。このようなシンボルに関する情報を管理するのがシンボルテーブルであり、このテーブルを参照することによって、同じシンボルで表されたものが同じものか異なるものかを知ることができる。

また、「構文解析木」とは、プログラムの構成を木構造で表現したものである。プログラムの解析を行う際にはまず構文解析が行われるが、その際ソースプロ

グラムをコンピュータが理解しやすくなるように木構造で表現したものを言う。

また、「コールグラフ」とは、プログラムを構成する各手続き（Ｃ言語では「関数」）の間の呼び出し関係を表したグラフである。このグラフのノードは手続きを表し、エッジは手続き間の呼び出し関係を表している。このコールグラフに基づいて、例えば図３のプログラム解析部２０$_1$がノードを丸印、エッジを矢印でグラフィック表示する。なお、コールグラフをグラフィック表示させると、手続き間の呼び出し関係が視覚的に把握できるだけでなく、プログラムの構造化の程度や内容の理解、あるいは望ましくない手続き呼び出しの検出が容易になる。

また、「フローグラフ」とは、ある一つの手続き内における制御の流れ（フロー）を表したグラフである。このグラフのノードは基本ブロック（連続した文の列からなり、制御が先頭の文に移された後は、途中で停止したり分岐したりせず、最後の文まで制御が到達するものを言う）を表し、エッジは基本ブロック間の制御のフローを表す。このフローグラフに基づいて、例えば図３のプログラム解析部２０$_2$がノードを丸印で、エッジを矢印でグラフィック表示する。

このフローグラフについても、その情報をグラフィック表示させると、その手続き内の制御フローが視覚的に把握できるだけでなく、プログラム中の望ましくない制御フローの検出が可能となる。例えば、制御が到達することのないデッドコードは、入口ノードからエッジを辿っていったとしても到達することのないノードであるが、入口ノードからのつながりがないので、容易に理解することができる。

また、「データフロー情報」とは、例えば、ポインタのエイリアス情報や、ある場所で定義された変数がどこで使用されるかといった定義－使用連鎖の情報などのデータの流れを解析した結果の情報である。このデータフロー情報等を利用して、例えばデータフローの異常検査を行うことが可能である。データフロー異常とは、データに対する不当なイベントの組み合わせである。すべてのデータは、定義した上で使用し、最後に未定義にするという一連の順序を守って使用しなければならないが、このような順序を守らない不当な組み合わせは、データフロー異常となる。

データフロー異常が存在しても、ソースコードをコンパイルすることは可能で

ある。しかし、実際にコンパイルしたプログラムを実行させているときに、異常なパスを通るとデータフロー異常が顕在化することがある。したがって、このような異常は、事前にソースコードの段階で除去しておくのが望ましい。データフロー異常の検出については、本出願人による特許出願（特願平９－３２４１５号および特願平９－３２４５２号）を参照することができる。

また、「プログラム依存グラフ」とは、プログラム中における制御の依存関係（関数同士の依存関係）と、データの依存関係（データを関数に入れた場合の依存関係）とをグラフで表現したものである。プログラム依存グラフが作成されると、例えば、プログラムの影響範囲解析が実行できる。影響範囲解析とは、プログラム中のある場所（一つの文、または一つの変数）を変更したときに、それがどこまで影響を与えるか、あるいは影響を与えないかを求める解析である。

また、「モジュール入出力情報」とは、プログラムを構成するモジュール（例えば、Ｃ言語であれば関数にあたる）の入力と出力の情報である。これは、引数や返り値での入出力に加え、入出力関数や大域変数を用いた入出力の情報も求めたものである。

また、「メトリクス情報」とは、ソフトウェアの数値化指標に関する情報である。本実施例のソフトウェア解析装置では、量的複雑さを表すメトリクスと、質的複雑さを表すメトリクスとを計測し、その情報を作成する。量的複雑さのメトリクスとしては、プログラムの物理的な記述量を計測するサイズ・メトリクスと、制御構造の複雑さを計測するサイクロマティック数との二種類を計測する。一方、質的複雑さのメトリクスとしては、モジュールの内容そのものを表す凝集度と結合度とを計測する。

また、「冗長度情報」とは、単一の手続きについて、出力に影響しない冗長な文に関する情報である。冗長な文は出力に影響しないので、ソースコードから削除しても、外部に対する出力に変化はない。したがって、この冗長度情報に従って、このような冗長な文を確認して対応策をとっておくことによって、意図しない動作の発生等を防止することが可能となる。

また、「保守用文書情報」とは、プログラムを保守する際に用いるドキュメント類である。すなわち、プログラム中で定義されている手続きや型、変数名など

の一覧や、それぞれの手続き等がどこに関連したものであるかを記載したものである。この「保守用文書情報」は、例えば、他人が作成したプログラムを理解するときに使われる。「保守用文書情報」を取得しておくことで、ハイパーテキスト方式で情報提供をすることができ、紙ベースのドキュメントと比べると、格段に便利になる。

図１のプログラム解析情報作成部１１は、図２に示すように、上述した各種のプログラム解析情報を順次階層化して作成する。このように階層化するのは、上層のプログラム解析情報を作成するときに、それより下層のプログラム解析情報を利用することによるものである。このとき、一つの層ごとのひとまとまりのプログラム解析情報の作成が終了した段階で、プログラム解析情報格納部１２がこれらの解析情報をトランザクションとして順次データベースに格納する。

例えば、コールグラフやフローグラフを作成するには、それより下層の構文解析木の情報が必要となる。また、データフロー情報を作成するには、それより下層の構文解析木、コールグラフおよびフローグラフの情報が必要となる。また、プログラム解析処理部１５による対話処理（これについては後述する）でデータフローの異常検査を行うときは、図２に階層的に示した各種のプログラム解析情報のうち、構文解析木、コールグラフ、フローグラフ、そしてデータフロー情報までが必要となる。

このように、上層のあるプログラム解析情報を作成する場合に、既に作成されて情報記録媒体１３に格納されている下層のプログラム解析情報を必要とする関係を、ここでは「階層化」と呼んでいる。本実施例では、プログラム解析情報作成部１１が図２の下層のプログラム解析情報から順番に作成していき、得られた解析情報をプログラム解析情報格納部１２が情報記録媒体１３に順次格納していく。このとき、上層のあるプログラム解析情報を作成するときに必要な下層のプログラム解析情報は、プログラム解析情報読出部１４が情報記録媒体１３から読み出して、それをプログラム解析情報作成部１１に供給する。

仮に、トランザクション機能を用いずに、バッチ処理によって全てのプログラム解析情報を作成し、それから一括して情報記録媒体１３に格納するようにすると、対象とするソースファイル１０のソースコードが特に大規模なプログラムの

場合には、ワークステーションやパーソナルコンピュータに搭載できるメモリの容量を超えたデータ量を扱うことになり、結果的には解析が不可能となってそれまでに行った処理が無駄になることがある。

これに対して、本実施例のソフトウェア解析装置では、プログラム解析情報作成部１１が作成したプログラム解析情報を、プログラム解析情報格納部１２が各階層の各解析情報の種類ごとに分割し、これらを順次ハードディスク等の情報記録媒体１３にデータベース化して格納するようにしている。

これにより、メモリ内に一度に蓄えられる解析情報量は、個々に作られる解析情報の量となり、コンピュータ上の限られたメモリ容量の制約から開放される。よって、特に大規模なプログラムを解析する場合であっても、解析の途中でメモリ不足などが発生することが少なくなり、プログラムの解析を最後まで確実に行うことができるようになる。

また、図２のデータ構造に示す各階層の各解析情報ごとに、それらの解析情報が作成された段階で逐次トランザクションとしてハードディスク等の情報記録媒体１３に格納することによって、メモリ不足以外の何らかの障害が発生した場合でも、各種のプログラム解析情報のうち、それまでに情報記録媒体１３に格納された部分は安全に保護される。

例えば、構文解析木の作成からプログラム依存グラフの作成まで数日程度を要する大規模なプログラムの解析を行おうとした場合に、途中で何らかの障害が発生してコンピュータがダウンし、メモリ上に展開されていたデータが破壊されたとしても、少なくとも既にデータベースとして情報記録媒体１３に格納された部分については、安全に保護される。

よって、次に作業を再開するときは、既にデータベースに格納されているプログラム解析情報については、改めて作成処理を行う必要はなく、それ以降の作成処理から再開すれば良くなる。なお、解析の中断後に再び解析を行うときは、どの段階から開始するのかをオペレータが装置に指示する。このように、本実施例によれば、中断される間際までに得られたプログラム解析情報を有効に利用して、重複した作業を回避することができる。

なお、本実施例では、データベースとしては、オブジェクト指向のデータベー

スソフトウェアを利用し、プログラム解析情報は、オブジェクト指向言語、例えばＣ＋＋などを用いて作成する。このようなデータベースを利用するのは、プログラム解析情報はグラフ構造など、複雑な構造と関連を持つものなので、リレーショナルデータベースを用いたのでは格納が困難なためである。また、Ｃ＋＋言語上でオブジェクトとして作成されたプログラム解析情報を、作成された構造のまま格納することができるので、便利である。

次に、図３を参照して、プログラム解析処理部１５の内部構成および動作について説明する。図３には、複数のプログラム解析部20_1，20_2…20_iが示されている（以下、これらをまとめて単にプログラム解析部２０とも記す）。これらのプログラム解析部２０は、ソースファイル１０のソースコード、およびプログラム解析情報格納部１２によってハードディスク等の情報記録媒体１３にデータベース化して格納されているプログラム解析情報に基づいて、種々のプログラム解析を実行する部分である。

各プログラム解析部20_1，20_2……20_iは、解析する種類に対応して設けられている。本実施例では、例えばプログラム解析部20_1は、コンピュータ画面上にコールグラフをグラフィック表示させるための解析処理を行う。これにより、コールグラフの内容を実際に視覚的に把握できるようになり、プログラム内の関数の呼出し関係の理解が容易となる。この場合、プログラム解析部20_1は、ソースコードを取り込むとともに、プログラム解析情報として、構文解析木およびコールグラフの情報を図１のプログラム解析情報読出部１４を介して情報記録媒体１３から読み出してくる。　また、プログラム解析部20_2は、コンピュータ画面上にフローグラフをグラフィック表示させるための解析処理を行う。これにより、フローグラフの内容を実際に視覚的に把握できるようになり、ある関数の内部における変数の流れの理解が容易となる。この場合、プログラム解析部20_2は、ソースコードを取り込むとともに、プログラム解析情報として、構文解析木およびフローグラフの情報を図１のプログラム解析情報読出部１４を介して情報記録媒体１３から読み出してくる。

また、プログラム解析部20_1は、プログラム解析情報格納部１２によって情報記録媒体１３に格納された各種のプログラム解析情報を用いて、バッチ処理で

解析することができる解析処理を行い、その結果をバッチ解析結果ファイルとして再び情報記録媒体１３に格納する。このバッチ処理中は、オペレータからの指示などは特に必要なく、本装置が自動的に解析を行う。バッチ解析結果ファイルには、例えばプログラムの規模に関連するメトリクス情報や冗長度情報、保守用文書情報などが含まれる。

このように、メトリクス情報、冗長度情報および保守用文書情報は、バッチ解析処理の結果生成されるが、バッチ解析処理の結果生成されるものはこれらに限られない。すなわち、必要に応じて種々のバッチ解析処理を行い、その結果、それぞれのバッチ解析処理に対応した種々の解析情報を生成するように、プログラム解析部20_iを設計することが可能である。

これらのプログラム解析部20_1，20_2……20_iは、情報記録媒体１３から必要なプログラム解析情報を取り出してきて、所定の解析処理を行い、その解析結果21_1，21_2……21_iをコンピュータ画面上にＧＵＩ（Graphical User Interface）22_1，22_2……22_iにより表示させる。なお、ＧＵＩ22_1，22_2……22_iは、解析処理の結果の内容を、オペレータがコンピュータ画面上で見ることができるように表示させるためのものである。

このとき、バッチ解析処理を行うプログラム解析部20_i以外では特に、オペレータは、この表示結果を見て、何らかの指示が必要な場合には、ＧＵＩ22_1，22_2…を介してプログラム解析部20_1，20_2…へ指示を送り、各プログラム解析部20_1，20_2…に再度解析処理を行わせ、その結果を表示させる。このような処理を繰り返しながら、プログラム解析を行っていく。このように、オペレータがＧＵＩ22_1，22_2…を介してコンピュータに指示を出し、それに対してコンピュータが解析処理を行うような操作を「対話処理」と呼ぶ。

プログラム解析部２０が実行する解析処理の他の例として、影響範囲解析がある。これは、プログラム解析情報として、構文解析木、コールグラフ、フローグラフ、プログラム依存グラフ、モジュール入出力情報を情報記録媒体１３から読み出して、プログラムのソースコードのある行が、他のどの行まで影響を及ぼしているかを解析する。

具体的には、プログラムのソースコードをコンピュータ画面上に表示し、そこ

である行を指定して解析を行うことにより、その指定された行がソースコードのどこに影響しているかを、例えば色を変えて表示する。そして、他の行からの影響も調べたいときは、その行を新たに指定して、それがどの行に影響しているかということを調べる。この解析により、オペレータは、例えば、プログラムの修正の影響が自分の意図通りであるかどうかを確認することができる。

また、プログラム解析部２０が行う解析処理の別の例としては、構造解析、データフロー異常解析などがある。本実施例では、データフロー異常解析は対話処理によって行う場合について説明したが、これをバッチ処理で行うことも可能である。

なお、図３において、ＧＵＩ連動部２３は、ＧＵＩ22_1，22_2……22_iのうち任意の数のＧＵＩをコンピュータ画面に表示させ、例えばあるＧＵＩ画面に表示されたソースプログラムのある行を指定すると、他のＧＵＩ画面に表示されたフローグラフの対応する部分が色違いで表示されるというように、複数のＧＵＩ画面を関連付けて表示するための部分である。これを用いることによって、プログラム内容の理解がより容易となる。

本実施例のソフトウェア解析装置では、プログラム解析情報作成部１１において、どの解析情報まで作成するかを、オペレータが任意に指定できるようにしてもよい。プログラムの開発中の段階では、このような要求がよく発生するが、このような指定をできるようにすると、例えば、コールグラフを表示するだけという場合には、それより上層の解析情報は必要ないので、その旨を指定しておくことにより、バッチ処理による解析処理をより高速化することができる。

次に、図４を参照して、本実施例によるソフトウェア解析装置の全体の動作について説明する。図４において、まずステップＳ１でオペレータが解析するソースコードを指示すると、ステップＳ２でそれに対応するプログラム解析情報のデータベースが情報記録媒体１３に存在するかどうかが調べられ、存在する場合には、ステップＳ３で当該プログラム解析情報がプログラム解析情報読出部１４によって読み出される。

そして、ステップＳ４でオペレータは、ソフトウェア解析装置に対してこれから作成するプログラム解析情報を指示する。例えば、最下層の解析情報からどの

解析情報まで作成するかなどを指示する。また、前回の作成途中で何らかの原因により処理が中断してしまい、今回その作成処理を再開する場合には、どの解析情報の作成から始めるかを指示する。

次に、ステップＳ５で、プログラム解析情報作成部１１が、オペレータから入力された解析情報の作成指示に従って、プログラム解析情報の作成を実行する。このとき、処理が進められていくごとに、作成されたプログラム解析情報がメモリ内に順次蓄えられていく。次のステップＳ６では、プログラム解析情報格納部１２が、作成されたプログラム解析情報を、上記メモリから情報記録媒体１３にデータベース化して格納する。

以上のようにしてプログラム解析を行うために必要なプログラム解析情報が作成されると、次にステップＳ７でオペレータは、これから行うプログラム解析処理の種類を装置に指示する。これに対応してプログラム解析処理部１５は、ステップＳ８で、図３のプログラム解析部20_1，20_2……20_iの何れかによりプログラム解析処理を実行する。

このプログラム解析が終わると、ステップＳ９で、他の種類のプログラム解析を行うか否かを判断し、行う場合はステップＳ７に戻り、行わない場合はステップＳ１０に進む。ステップＳ１０では、他のソースコードの解析を行うか否かを判断し、行う場合はステップＳ１に戻り、行わない場合は処理を終了する。

次に、本発明のコンピュータ読み取り可能な記録媒体について説明する。本発明のソフトウェア解析装置は、上述したコンピュータが備えるＲＯＭやＲＡＭ等に記憶されたプログラムに従って動作することによって実現されるが、外部からプログラムを読み込んだコンピュータが、そのプログラムに従って動作することによって実現されるようにしても良い。したがって、このようプログラムを記録した記録媒体、例えばフロッピーディスク、ＣＤ－ＲＯＭ、光磁気ディスク、磁気テープ、半導体記憶装置などをコンピュータに装着し、当該プログラムを読み込ませることによって、コンピュータ上で本発明を実施することが可能となる。

なお、本発明は、上記実施例に限定されるものではなく、その要旨の範囲内で種々の変更が可能である。例えば、上記実施例では、各層の１つの種類の解析情報が作成された時点で、得られたプログラム解析情報を逐次データベースに格納

しているが、例えば各層の解析情報の作成が終了した時点や、メモリ上に所定の情報量が蓄積された時点で逐次データベースに格納するようにしても良い。また、所定の時間が経過するごとに、そのときまでに得られたプログラム解析情報を逐次データベースに格納するようにしても良い。

以上説明したように、本発明によれば、プログラム解析情報作成手段によって作成されたプログラム解析情報を、適当な単位もしくは適当なタイミングごとに順次所定のデータ記録媒体に格納していくようにしたので、大規模なプログラムに対してプログラム解析情報を得ようとした場合に、コンピュータに実装できるメモリ容量に制限があっても、解析の途中でメモリ不足などが発生することもなく、そのプログラム解析情報を安全に得ることができる。

また、何らかの障害が発生して解析処理が中断されてしまった場合でも、それまでにデータ記録媒体に格納された情報は安全に保護されるので、次に作業を再開するときは、前回中断されたところから開始すれば良く、重複した処理を回避して作業の効率を格段に向上させることができる。

また、解析情報の作成後に何らかの解析処理を対話的に行う場合、既にデータベースに格納されている解析情報をそのまま利用できるので、その解析情報作成のための時間を省くことができ、何らかの解析処理を行う度に解析情報を最初から作成していた従来と比べて、オペレータの待ち時間を減らすことができる。

【特許請求の範囲】（What is claimed is:）

1．コンピュータプログラムの解析に必要なプログラム解析情報を自動的に作成するプログラム解析情報作成手段と、

上記プログラム解析情報作成手段によって作成された上記プログラム解析情報を、適当な単位もしくは適当なタイミングごとに所定のデータ記録媒体に順次格納するプログラム解析情報格納手段と、

上記データ記録媒体から上記プログラム解析情報を読み出してプログラム解析を実行するプログラム解析手段とを備えたことを特徴とするソフトウェア解析装置。

2．上記プログラム解析手段は、上記データ記録媒体から上記プログラム解析情報を読み出して、オペレータとの間の対話的処理によりプログラム解析を実行することを特徴とする請求項1に記載のソフトウェア解析装置。

3．上記プログラム解析手段は、上記データ記録媒体から上記プログラム解析情報を読み出して、バッチ的処理によりプログラム解析を実行することを特徴とする請求項1に記載のソフトウェア解析装置。

4．上記プログラム解析手段は、上記バッチ的処理によりメトリクス情報、冗長度情報、データフロー異常情報、保守用文書情報のうち少なくとも一つを作成することを特徴とする請求項3に記載のソフトウェア解析装置。

5．上記プログラム解析情報作成手段は、複数種類のプログラム解析情報を順次作成し、上記プログラム解析情報格納手段は、個々の種類のプログラム解析情報が作成されるごとに上記データ記録媒体に格納することを特徴とする請求項1に記載のソフトウェア解析装置。

6．上記複数種類のプログラム解析情報のうちどの範囲を対象として作成するかを指示するための範囲指示手段を備えたことを特徴とする請求項5に記載のソフトウェア解析装置。

7．上記プログラム解析情報格納手段は、上記プログラム解析情報をデータベース化して上記データ記録媒体に格納することを特徴とする請求項1に記載のソフトウェア解析装置。

8．上記プログラム解析情報には、

上記コンピュータプログラムのソースコードに基づいて作成される構文解析木、

上記コンピュータプログラムのソースコードに用いられているシンボルの意味を表すシンボルテーブル、

上記構文解析木に基づいて作成されるコールグラフあるいはフローグラフ、

上記構文解析木、シンボルテーブル、フローグラフおよびコールグラフに基づいて作成されるデータフロー情報、

上記構文解析木、シンボルテーブル、コールグラフ、フローグラフおよびデータフロー情報に基づいて作成されるプログラム依存グラフあるいはモジュール入出力情報、

のうち少なくとも一つが含まれていることを特徴とする請求項1に記載のソフトウェア解析装置。

9．上記プログラム解析情報作成手段は、上記構文解析木およびシンボルテーブル、コールグラフおよびフローグラフ、データフロー情報、並びにプログラム依存グラフおよびモジュール入出力情報を、この順序で作成することを特徴とする請求項8に記載のソフトウェア解析装置。

１０．上記プログラム解析情報格納手段は、上記プログラム解析情報作成手段によって上記構文解析木およびシンボルテーブル、コールグラフ、フローグラフ、データフロー情報、プログラム依存グラフ、モジュール入出力情報が作成されるごとに、個々のプログラム解析情報を上記データ記録媒体に順次格納することを特徴とする請求項9に記載のソフトウェア解析装置。

１１．コンピュータプログラムの解析に必要なプログラム解析情報を作成し、作成された上記プログラム解析情報を解析の目的別に階層的にデータベースに登録するようにし、上記コンピュータプログラムの解析を行うときは、解析する目的に応じて、所定の階層に既に登録されている上記プログラム解析情報を読み出して解析を実行するようにしたことを特徴とするソフトウェア解析装置。

１２．上記データベースは、オブジェクト指向型のデータベースであることを特徴とする請求項１１に記載のソフトウェア解析装置。

１３．コンピュータプログラムの解析に必要なプログラム解析情報を自動的に

作成するプログラム解析情報作成ステップと、

上記プログラム解析情報作成ステップにて作成された上記プログラム解析情報を、適当な単位もしくは適当なタイミングごとに所定のデータ記録媒体に順次格納するプログラム解析情報格納ステップと、

上記データ記録媒体から上記プログラム解析情報を読み出してプログラム解析を実行するプログラム解析ステップとを有することを特徴とするソフトウェア解析方法。

１４．上記プログラム解析ステップでは、上記データ記録媒体から上記プログラム解析情報を読み出して、オペレータとの間の対話的処理によりプログラム解析を実行することを特徴とする請求項１３に記載のソフトウェア解析方法。

１５．上記プログラム解析ステップでは、上記データ記録媒体から上記プログラム解析情報を読み出して、バッチ的処理によりプログラム解析を実行することを特徴とする請求項１３に記載のソフトウェア解析方法。

１６．上記プログラム解析ステップでは、上記バッチ的処理によりメトリクス情報、冗長度情報、データフロー異常情報、保守用文書情報のうち少なくとも一つを作成することことを特徴とする請求項１５に記載のソフトウェア解析方法。

１７．上記プログラム解析情報作成ステップでは、複数種類のプログラム解析情報を順次作成し、上記プログラム解析情報格納ステップでは、個々の種類のプログラム解析情報が作成されるごとに上記データ記録媒体に格納することを特徴とする請求項１３に記載のソフトウェア解析方法。

１８．上記プログラム解析情報格納ステップでは、上記プログラム解析情報をデータベース化して上記データ記録媒体に格納することを特徴とする請求項１３に記載のソフトウェア解析方法。

１９．上記プログラム解析情報には、

上記コンピュータプログラムのソースコードに基づいて作成される構文解析木、

上記コンピュータプログラムのソースコードに用いられているシンボルの意味を表すシンボルテーブル、

上記構文解析木に基づいて作成されるコールグラフあるいはフローグラフ、

上記構文解析木、シンボルテーブル、フローグラフおよびコールグラフに基づいて作成されるデータフロー情報、

上記構文解析木、シンボルテーブル、コールグラフ、フローグラフおよびデータフロー情報に基づいて作成されるプログラム依存グラフあるいはモジュール入出力情報、

のうち少なくとも一つが含まれていることを特徴とする請求項１３に記載のソフトウェア解析方法。

２０．上記プログラム解析情報作成ステップでは、上記構文解析木およびシンボルテーブル、コールグラフおよびフローグラフ、データフロー情報、並びにプログラム依存グラフおよびモジュール入出力情報を、この順序で作成することを特徴とする請求項１９に記載のソフトウェア解析方法。

２１．上記プログラム解析情報格納ステップでは、上記プログラム解析情報作成ステップにて上記構文解析木およびシンボルテーブル、コールグラフ、フローグラフ、データフロー情報、プログラム依存グラフ、モジュール入出力情報が作成されるごとに、個々のプログラム解析情報を上記データ記録媒体に順次格納することを特徴とする請求項２０に記載のソフトウェア解析方法。

２２．コンピュータプログラムの解析に必要なプログラム解析情報を作成し、作成された上記プログラム解析情報を解析の目的別に階層的にデータベースに登録するようにし、上記コンピュータプログラムの解析を行うときは、解析する目的に応じて、所定の階層に既に登録されている上記プログラム解析情報を読み出して解析を実行するようにしたことを特徴とするソフトウェア解析方法。

２３．上記データベースは、オブジェクト指向型のデータベースであることを特徴とする請求項２１に記載のソフトウェア解析方法。

２４．コンピュータプログラムの解析に必要なプログラム解析情報を自動的に作成するプログラム解析情報作成機能と、

上記プログラム解析情報作成機能によって作成された上記プログラム解析情報を、適当な単位もしくは適当なタイミングごとに所定のデータ記録媒体に順次格納するプログラム解析情報格納機能と、

上記データ記録媒体から上記プログラム解析情報を読み出してプログラム解析

を実行するプログラム解析機能と、をコンピュータに実行させるためのプログラムを記録したことを特徴とするコンピュータ読み取り可能な記録媒体。

２５．上記プログラム解析機能は、上記データ記録媒体から上記プログラム解析情報を読み出して、オペレータとの間の対話的処理によりプログラム解析を実行することを特徴とする請求項２４に記載のコンピュータ読み取り可能な記録媒体。

２６．上記プログラム解析機能は、上記データ記録媒体から上記プログラム解析情報を読み出して、バッチ的処理によりプログラム解析を実行することを特徴とする請求項２５に記載のコンピュータ読み取り可能な記録媒体。

２７．上記プログラム解析機能は、上記バッチ的処理によりメトリクス情報、冗長度情報、データフロー異常情報、保守用文書情報のうち少なくとも一つを作成することを特徴とする請求項２６に記載のコンピュータ読み取り可能な記録媒体。

２８．上記プログラム解析情報作成機能は、複数種類のプログラム解析情報を順次作成し、上記プログラム解析情報格納機能は、個々の種類のプログラム解析情報が作成されるごとに上記データ記録媒体に格納することを特徴とする請求項２４に記載のコンピュータ読み取り可能な記録媒体。

２９．上記プログラム解析情報格納機能は、上記プログラム解析情報をデータベース化して上記データ記録媒体に格納することを特徴とする請求項２４に記載のコンピュータ読み取り可能な記録媒体。

３０．上記プログラム解析情報には、

上記コンピュータプログラムのソースコードに基づいて作成される構文解析木、

上記コンピュータプログラムのソースコードに用いられているシンボルの意味を表すシンボルテーブル、

上記構文解析木に基づいて作成されるコールグラフあるいはフローグラフ、

上記構文解析木、シンボルテーブル、フローグラフおよびコールグラフに基づいて作成されるデータフロー情報、

上記構文解析木、シンボルテーブル、コールグラフ、フローグラフおよびデー

タフロー情報に基づいて作成されるプログラム依存グラフあるいはモジュール入出力情報、
のうち少なくとも一つが含まれていることを特徴とする請求項２４に記載のコンピュータ読み取り可能な記録媒体。

３１．上記プログラム解析情報作成機能は、上記構文解析木およびシンボルテーブル、コールグラフおよびフローグラフ、データフロー情報、並びにプログラム依存グラフおよびモジュール入出力情報を、この順序で作成することを特徴とする請求項３０に記載のコンピュータ読み取り可能な記録媒体。

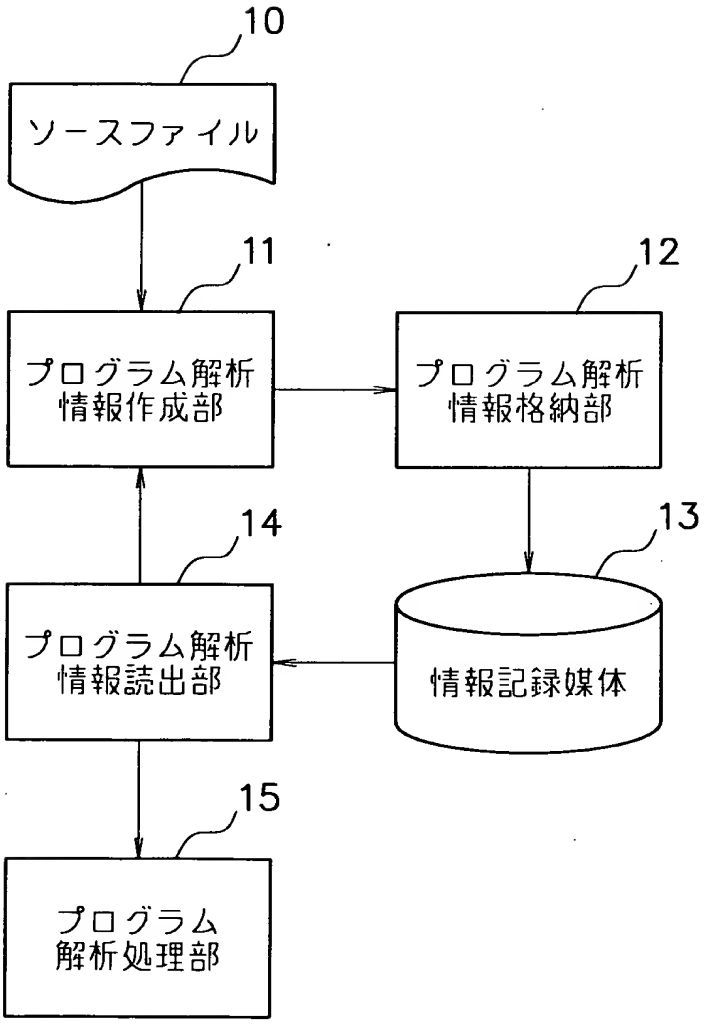
３２．上記プログラム解析情報格納機能は、上記プログラム解析情報作成機能によって上記構文解析木およびシンボルテーブル、コールグラフ、フローグラフ、データフロー情報、プログラム依存グラフ、モジュール入出力情報が作成されるごとに、個々のプログラム解析情報を上記データ記録媒体に順次格納することを特徴とする請求項３１に記載のコンピュータ読み取り可能な記録媒体。

３３．コンピュータプログラムの解析に必要なプログラム解析情報を作成し、作成された上記プログラム解析情報を解析の目的別に階層的にデータベースに登録するようにし、上記コンピュータプログラムの解析を行うときは、解析する目的に応じて、所定の階層に既に登録されている上記プログラム解析情報を読み出して解析を実行する機能をコンピュータに実現させるためのプログラムを記録したことを特徴とするコンピュータ読み取り可能な記録媒体。

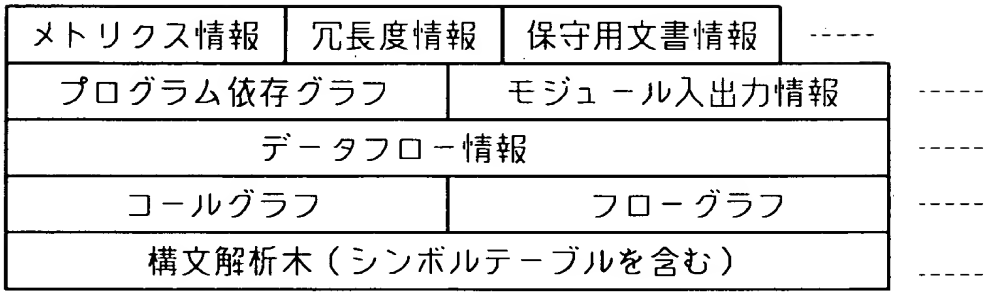
３４．上記データベースは、オブジェクト指向型のデータベースであることを特徴とする請求項３３に記載のコンピュータ読み取り可能な記録媒体。

【書類名】 図面

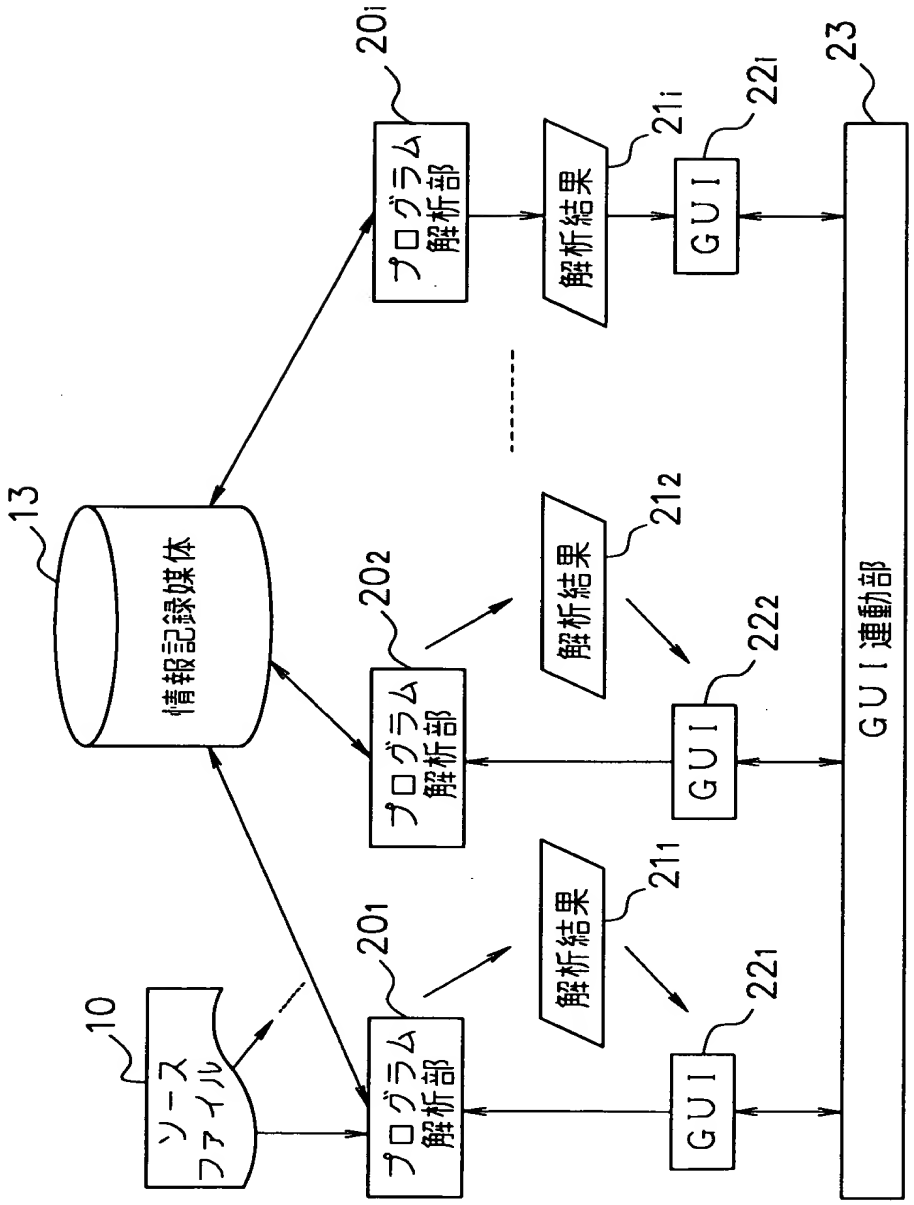
【図1】


10
ソースファイル
11
プログラム解析
情報作成部
12
プログラム解析
情報格納部
13
情報記録媒体
14
プログラム解析
情報読出部
15
プログラム
解析処理部

【図2】



【図3】



【図4】